99 82632

B2

(12) UK Patent Application (19) GB (11) 2 308 777 (13) A

(43) Date of A Publication 02.07.1997

(21) Application No 9526600.3

(22) Date of Filing 28.12.1995

(71) Applicant(s)
Nokia Telecommunications OY

(Incorporated in Finland)

Mäkkylän puistotie 1, FIN-02600 Espoo, Finland

(72) Inventor(s)
Graham French
David Bending

(74) Agent and/or Address for Service
Frank B Dehn & Co
179 Queen Victoria Street, LONDON, EC4V 4EL, United Kingdom

(51) INT CL[6]
H04Q 3/00

(52) UK CL (Edition O)
H4K KF42

(56) Documents Cited
WO 94/06232 A1

(58) Field of Search
UK CL (Edition O) **H4K KF42**
INT CL[6] **H04Q 3/00**
ONLINE : WPI

(54) Telecommunications network management

(57) A telecommunications network management system has a database DB including information about the network, the information being in the form of objects that relate to the network elements to be managed by one or more operatives running applications APPLN at system parts PP1 ... PPN peripheral to a core part CP. In order to reduce the required bandwidth between applications APPLN and the core part CP, an object selection procedure using a set of Attribute Value Assertions (AVAs) is effected at a selection point so that only certain objects will be selected. Each AVA comprises an attribute identifier, a value, and an operation. For each AVA, the attribute identifier is used to obtain the value of this attribute for the object subject to selection, and the AVA operation is applied to this value and the value in the AVA to obtain an intermediate result. The intermediate results for the set of AVAs are logically combined to obtain a final result, and the object is selected or not selected on the basis of this final result. A similar operation using AVAs may be used for filtering the passage of events.

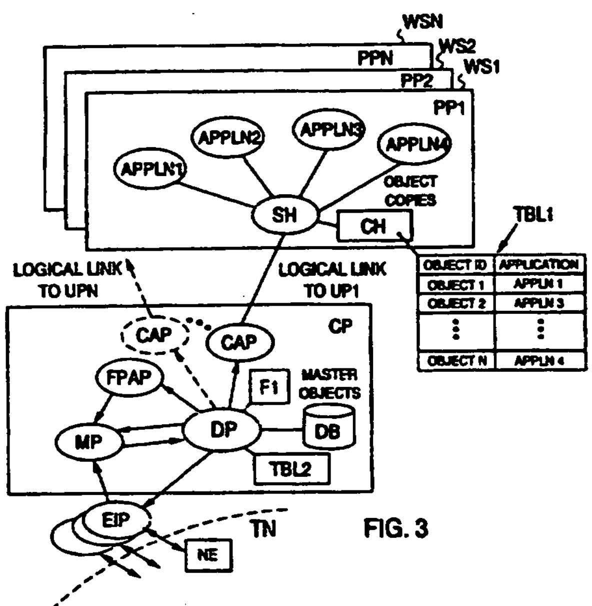

FIG. 3

GB 2 308 777 A

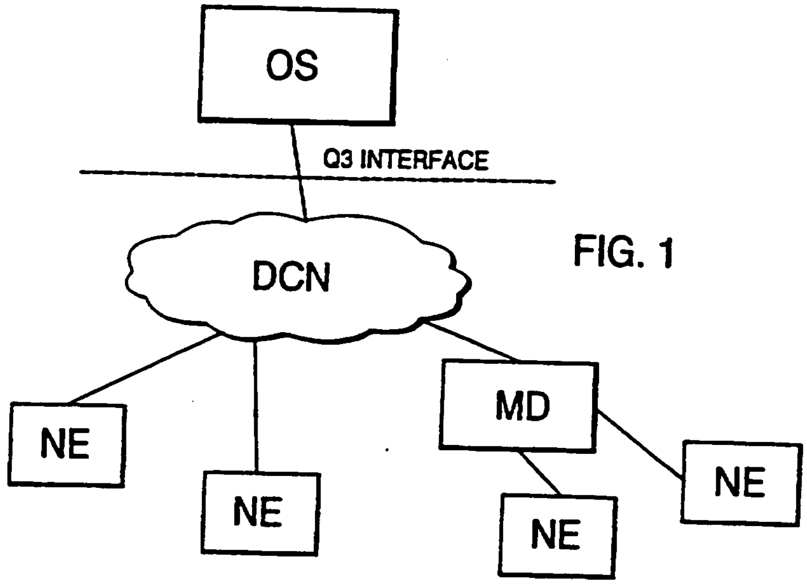
OS
Q3 INTERFACE
DCN
NE
NE
MD
NE
NE

FIG. 1

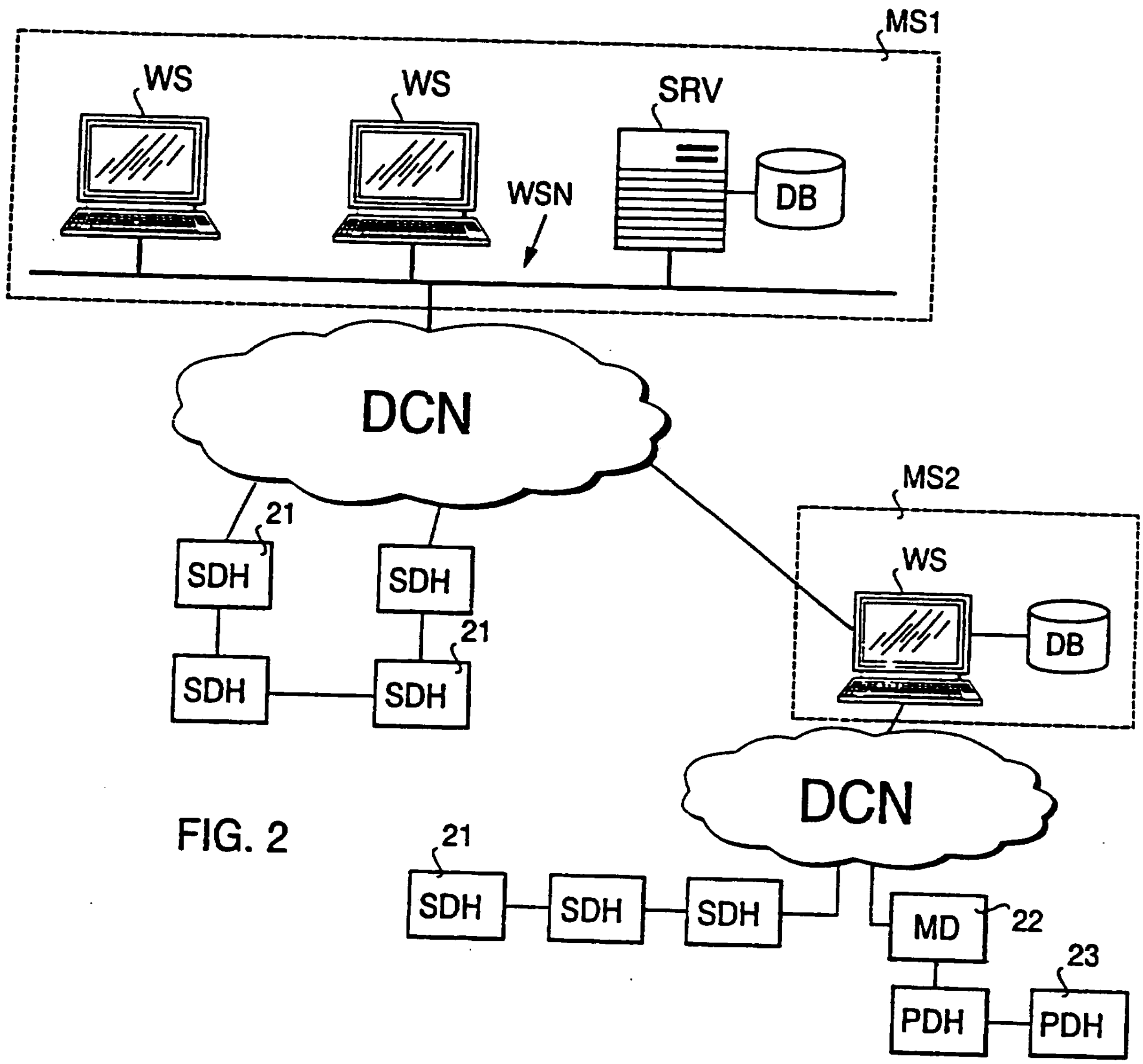
MS1
WS
WS
SRV
WSN
DB
DCN
21
SDH
SDH
SDH
21
SDH
MS2
WS
DB
DCN
21
SDH
SDH
SDH
MD
22
23
PDH
PDH

FIG. 2

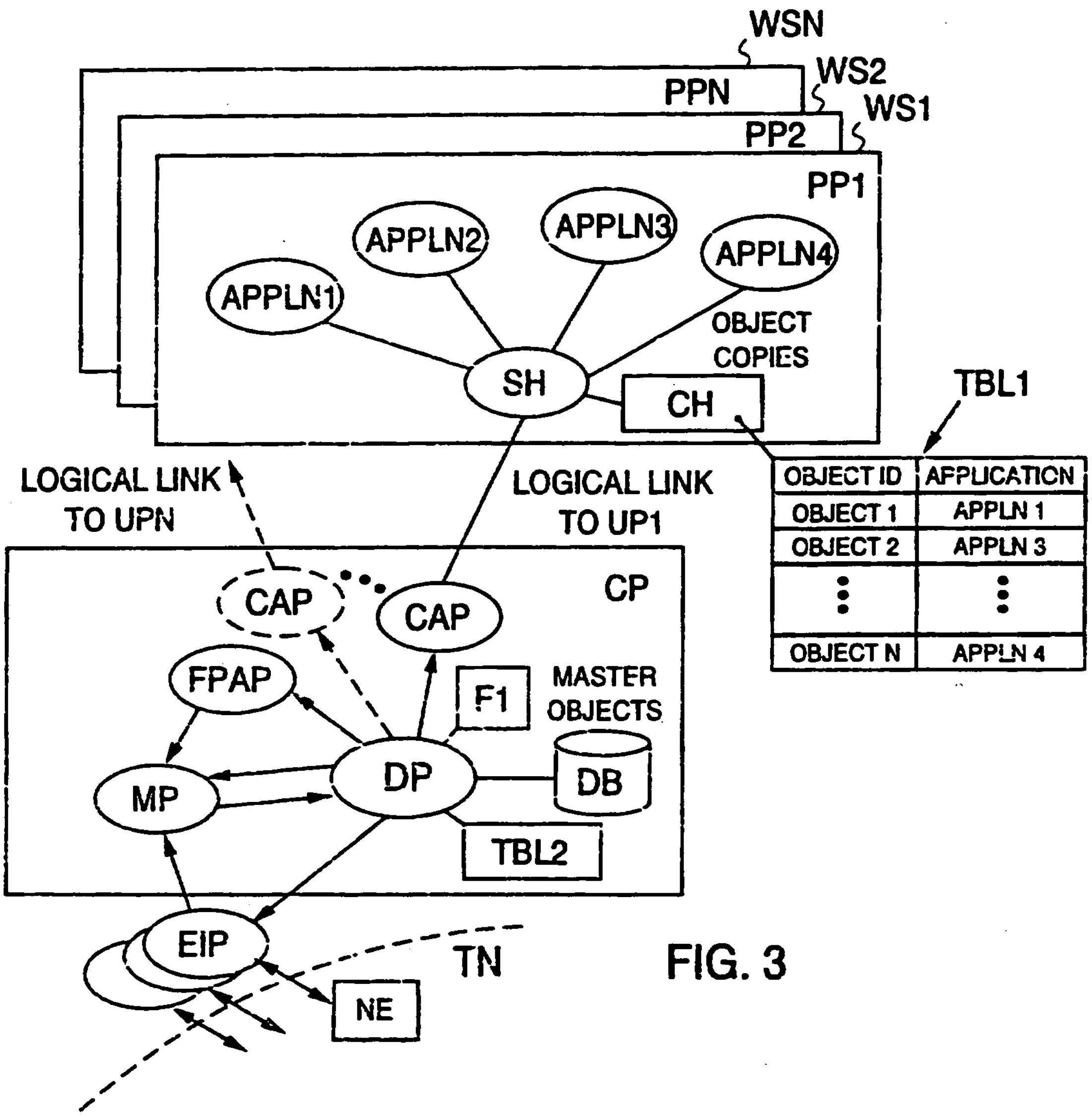
WSN
WS2
WS1
PPN
PP2
PP1
APPLN1
APPLN2
APPLN3
APPLN4
OBJECT COPIES
SH
CH
TBL1
OBJECT ID | APPLICATION
OBJECT 1 | APPLN 1
OBJECT 2 | APPLN 3
OBJECT N | APPLN 4
LOGICAL LINK TO UPN
LOGICAL LINK TO UP1
CP
CAP
CAP
FPAP
F1
MASTER OBJECTS
MP
DP
DB
TBL2
EIP
TN
NE

FIG. 3

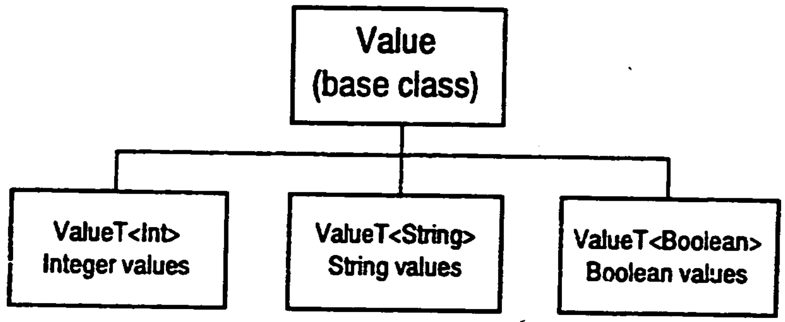

FIG. 4

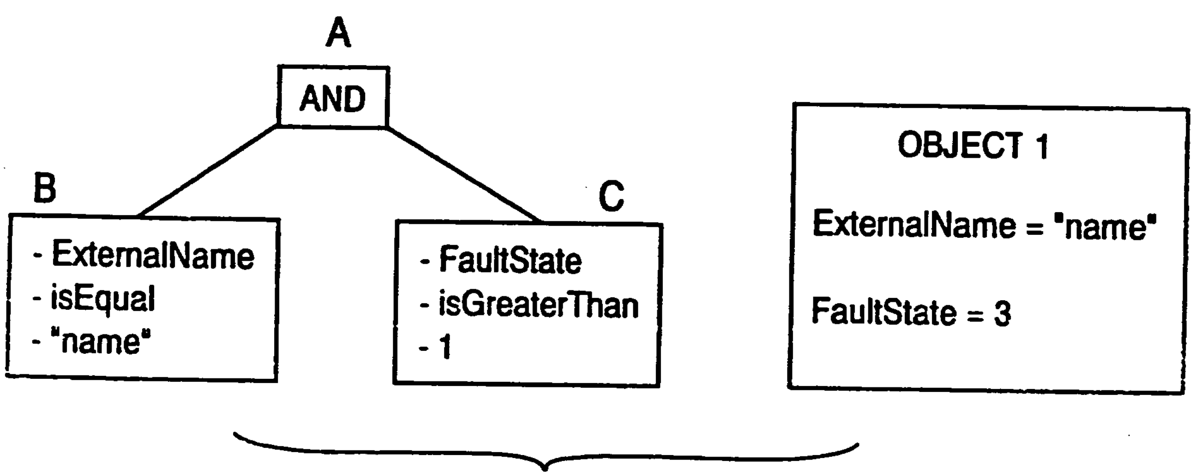

FIG. 7

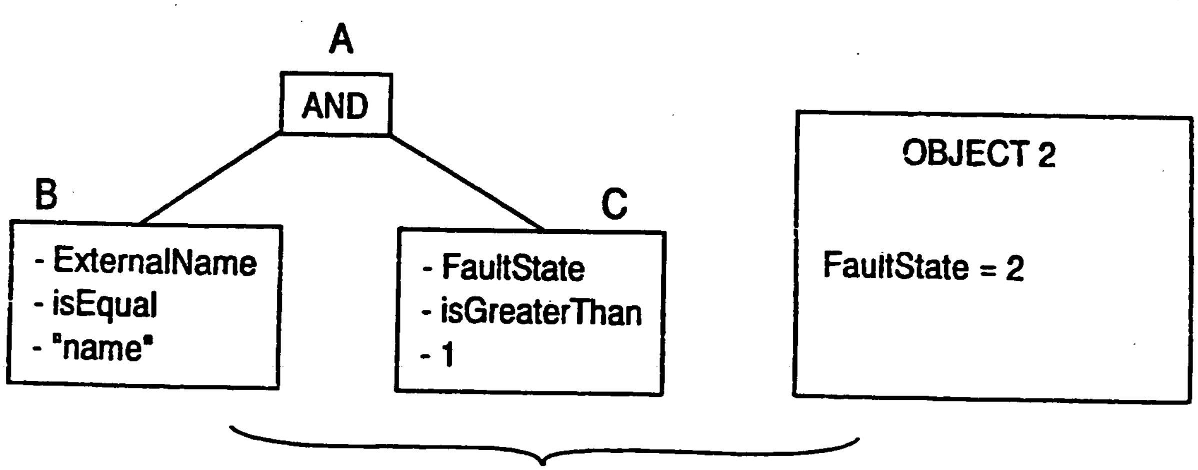

FIG. 8

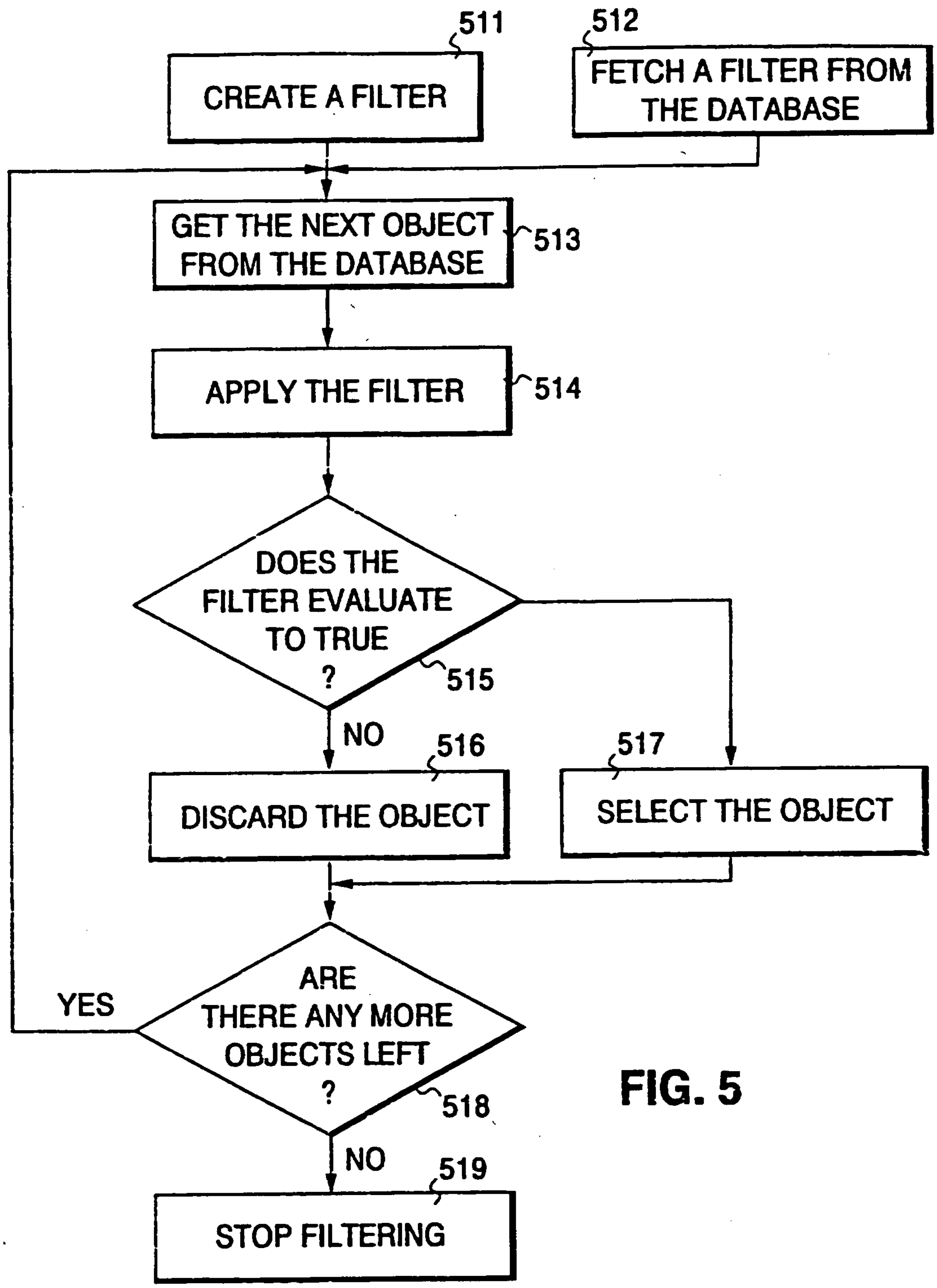
511
CREATE A FILTER
512
FETCH A FILTER FROM THE DATABASE
GET THE NEXT OBJECT FROM THE DATABASE
513
APPLY THE FILTER
514
DOES THE FILTER EVALUATE TO TRUE ?
515
NO
516
DISCARD THE OBJECT
517
SELECT THE OBJECT
ARE THERE ANY MORE OBJECTS LEFT ?
YES
518
NO
519
STOP FILTERING

FIG. 5

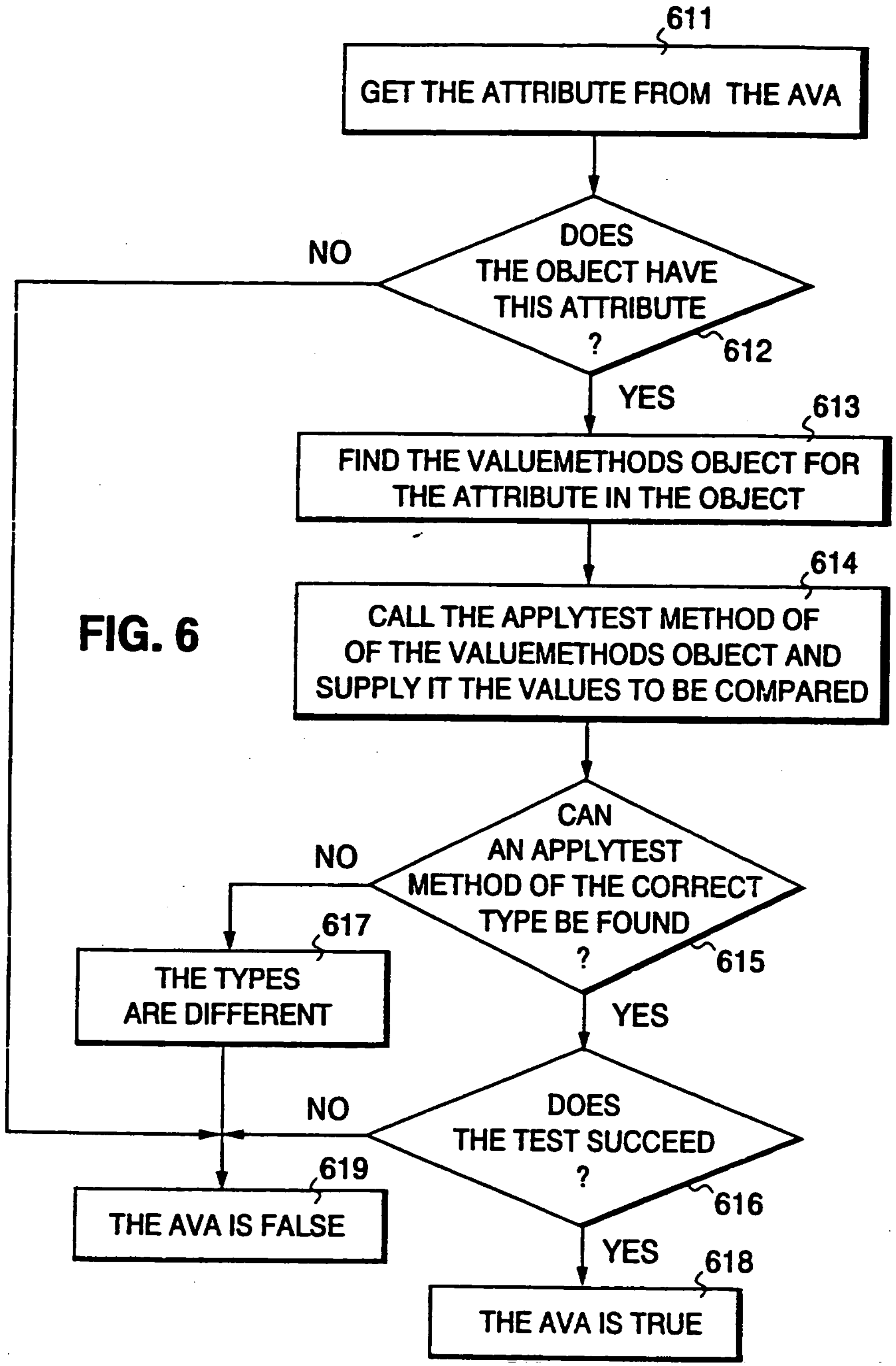

FIG. 6

NSDOCID: <GB 2308777A | >

Telecommunications network management method

The present invention relates to a management method according to the preamble of the attached claim 1 for managing a telecommunications network. The telecommunications network to be managed may be e.g. a SDH (Synchronous Digital Hierarchy) network, a PDH (Plesiochronous Digital Hierarchy) network, or a combination of such networks.

The basic situation in network management is usually such that an operator managing a telecommunication networks, e.g. a telephone company, has a plurality of customers (i.e. network users) in addition to the physical network. The operator sells the customers various services that utilize the network. (A public network will be used herein as an example; in principle, however, the same description applies to a private operator managing e.g. an organization network). To meet customers' data transmission requirements in the physical network, the operator utilizes a number of facilities or operative processes for the provision of customer services. These operative processes can be divided into groups in accordance with the functions for which they are intended:

- Service Provisioning taking care of the performance of customer services, including e.g. invoicing customers for services.

- Operation & Maintenance for keeping the network operative to allow the usage of customer services. One of the most important functions in this respect is the supervision and repair of network faults.

- Planning & Development, the function of which is to develop network operation so as to better meet

BNSDOCID: <GB 2308777A I >

customers' needs and to increase the overall profitability of the operator enterprise.

As appears from the above, network management takes place on several different levels, depending on the extent to which the functions to be performed on a specific level are associated with the overall management of the operator enterprise. The management of a telecommunications network is generally divided into four different levels, which are from bottom to top as follows:

- network element management layer,
- network management layer,
- service management layer, and
- business management layer.

This division is used e.g. in the ITU-T (the former CCITT) recommendation M.3010, which specifies a kind of framework for the management architecture of a telecommunications network. The bottom layer below the above four layers is the equipment itself; these equipments are managed by installation and field engineering tools.

The network element management layer means the management of an individual network element (such as a multiplexer or a cross-connection device) as a separate component without simultaneously paying attention to the condition of the network or other network elements. The majority of so called "network management" systems commercially available today are actually network element management systems within this layer.

The network management layer is concerned with the management of the entire telecommunications network, such as overall management of network connections. One example is the creation of connections and the end-to-end supervision of their condition.

This means that e.g. alarms detected on equipment are not just displayed against that equipment, but they are also propagated to show what services (paths and circuits) are affected by the fault, if any. The present invention is positioned in this layer.

As distinct from the above, the service management layer is not concerned with technical network management. It takes care of e.g. customer data, supervision of services provided to customers, invoicing for services, and considering needs for services of different types.

The business management layer is used to monitor and plan the business activities and economy of the entire enterprise, resulting in decisions affecting the lower levels.

At present, network management systems are changing into systems that manage the telecommunications network as a whole, whereas conventional management systems have handled only the remote control of transmission equipment, especially monitoring alarms produced by the equipment. In conventional network management methods, configuration changes, such as creation of new end-to-end connections, have been laborious and time-consuming, as the end result consists of several configuration events the prerequisite of which is that the maintenance staff of the network first gets an overall view of the situation and then decides on configuration changes required in individual network elements. In new network management systems, on the contrary, an overall view of the network and its condition is produced within the system, and the system itself gives the required configuration commands to each transmission equipment. As a consequence, all configuration changes can be performed significantly more

BNSDOCID: <GB 2308777A I >

rapidly than previously. Such developments have been accelerated by the freeing of competition in the field of telecommunications.

The above-mentioned recommendation M.3010 specifies the management architecture as shown in Figure 1. The architecture basically consists of one or more operations systems OS connected to a data communication network DCN communicating with an actual telecommunications network which is to be managed and which includes the network elements NE managed. It is to be noted that the connections of the data communications network and those of the telecommunications network are logically distinct, although they can be implemented physically in one and the same cable. Logically, there are thus two networks: (a) a network providing services to customers, and (b) a network maintaining the service provisioning network. The management of certain transmission equipments (network elements) further requires a separate Mediation Device MD, which mainly acts as a protocol converter between a Q3 interface complying with the recommendations and transmission equipments that do not understand the protocol applied in the interface but use their own proprietary protocol. New SDH equipment, for instance, can be connected directly to the Q3 interface, whereas older PDH equipment requires a Mediation Device.

In practice, a management network for a combined SDH and PDH network may be e.g. such as shown in Figure 2. Users (network operator staff) sitting at the operation centre use network management work stations WS connected to a separate local area network WSN, which may be e.g. an Ethernet network. The management system is typically distributed in several computers of the local area network, one of the

INSDOCID: <GB___2308777A__I_>

computers being a dedicated application server SRV having a database DB containing information necessary for managing the network. In its practical embodiment, the local area network further comprises e.g. necessary back-up devices (like DAT drives or mirrored disks) and event-logging printers (not shown).

The management system is connected via the above-mentioned Q3 interface e.g. to the SDH network. A variety of alternatives have been defined for the Q3 interface, so that the interface may be e.g. an X.25 type packet switched interface or an Ethernet LAN interface. (The packet switched interface is useful if the operator in charge of the network management also otherwise uses a packet switched network.) In practice, control channels between the SDH network elements 21 are established in the overhead bytes of the STM-N signal (N = 1, 4, 16), so that control signals between SDH equipments propagate with the payload signal (that is, also in the same physical network). Such control channels established in the overhead bytes are called Embedded Control Channels, and they are formed e.g. in the STM-1 frame by the section overhead bytes D1 to D12.

PDH equipments, on the contrary, need manufacturer-specific solutions, wherefore they have to be connected to the management system through a separate mediation device 22.

The management system may also be hierarchical so that different geographical areas have their own smaller management systems that together form an integral management network. For instance, a management system covering one country may be divided geographically into smaller management systems operating in different parts of the country. Each smaller management system takes care of the management

INSDOCID: <GB 2308777A I >

of the network portion in the concerned geographical area. In the example of Figure 2, management systems MS1 and MS2 geographically apart from each other form together a single common management system and management network.

Network management standards are nowadays largely based on so-called object-oriented descriptions, though the standards do not require the use of this technique. Objects are data structures in a network management system, which describe the functions and state of a network component. An object is thus an element having certain attributes ("I am like this") and certain operations ("I can do these things"). (In the object-oriented approach, objects with the same data structure (attributes) and the same behaviour (operations) are grouped into a class. A specific implementation of an operation is called a method and each object is said to be an instance of its class.) A typical object is e.g. a cross-connection device with certain attributes (cross-connections that are active) and certain methods (e.g. make cross-connection and release cross-connection).

In a telecommunications network management system, objects can be physical ones or logical ones. Physical objects are elements that form part of the physical network. Such objects are e.g. the above-mentioned network elements (a network element is any piece of telecommunication equipment that works as a single unit to provide telecommunication functions) or physical connections (such as optical fibres, twisted pair cables, coaxial cables, radio links or satellite links). Logical objects are logical entities that do not form a single piece of the physical network. Such objects are e.g. paths and circuits. (A path is a connection of a fixed bit rate and format between two

physical interfaces within the network. A circuit is a connection set up for a customer, between two physical interfaces on the boundary of the network. Thus, a circuit usually comprises several consecutive paths.)

A network object may have a number of different attributes. Some attributes (such as "fault state") are used by several different types of object. In addition, for some types of network object (such as a route), it is convenient to define an attribute which consists of a collection of other attributes. Typical attributes are e.g. "availability status", "fault state" and "operational state". The attributes have different possible values, e.g. "fault state" can have values:

- OK. There are no problems.

- Warning. There are outstanding faults, but these do not effect services.

- Degraded. Some or all of the services provided by the object are degraded.

- Failed. All the services provided by the object are lost.

- Unknown. The fault state of the object is unknown.

The "operational state", in turn, can have e.g. two different values:

- Enabled. The object can operate, either completely or in part.

- Disabled. The object cannot operate at all.

Thus, objects are used to represent the fundamental items with which the management system works, e.g. network elements, paths and users. To be able to examine the objects and to perform operations on them, the management system needs to obtain copies of these objects from the persistent storage. The objects can be fetched by their internal unique

identifier if this is known, or by performing a search of the objects using search criteria which define a group of conditions which an object must match to be selected.

Also within the network management system, events are passed around to various processes. An event may originate from e.g. from the network to be managed or it may occur as a result of the user actions, e.g. when the user gives a command from the workstation. To contain the number of events being passed, they are only passed on at each stage if some process in the system has indicated that it wishes to receive them. This wish to receive certain events is indicated by registering filters which specify the criteria which an event must match to be selected and hence passed on. (The selection mechanism using filters is identical to that used for selecting objects in searches.)

Thus, filtering/selecting is a way of specifying the information you are interested in.

In a network management system the criteria which can be used for filtering (and the attributes searched for) evolve over time, as different objects are included in the system (as a result of e.g. changes in the network to be managed). Therefore, the problem relating to filtering is how to get a method which allows new type of data and new type of objects to be added easily in the system, without effecting the ability to apply all filters to all objects and all events. The possibility to apply all filters to all objects should remain even if the objects do not have the attributes which are being filtered upon.

These goals are achieved by means of the method according to the invention, which is, in its first embodiment characterized by what is set forth in the

characterizing portion of the attached claim 1, and in its second embodiment by what is set forth in the characterizing portion of the attached claim 4.

The idea of the present invention is to find at a selection/filtering point of the system the value of an attribute in an object/event, apply a predetermined stored operation to that value and another value which is stored in the system, and evaluating the result. The above steps are repeated as many times as the stored filtering information shows and the results are then combined, using the stored filtering information, to get a final result for the filter. The final result determines whether the event is filtered (or the object selected).

Thanks to the solution according to the present invention, filtering can be applied to any object without any special knowledge of the type or attributes of the object to be filtered. The method is also generic, extensible and typesafe, i.e. it can use the comparisons appropriate for the attribute being filtered and copes with objects not containing that attribute.

Below, the invention and its preferred embodiments will be described in greater detail with reference to the examples of figures 4 to 6 of the attached drawings, in which

Figure 1 illustrates a telecommunications network management architecture,

Figure 2 shows an example of a management network for a combined SDH and PDH network,

Figure 3 illustrates the functional architecture of the management system according to the present invention in its basic form;

Figure 4 illustrates implementation of generic values,

Figure 5 is a flow chart describing how a filter is applied to searching for objects from the database,

Figure 6 is a flow chart which illustrates evaluating an AVA.

Figure 7 illustrates the method according to the invention when a filter is applied to an object which it matches, and

Figure 8 illustrates the method according to the invention when a filter is applied to an object which it does not match.

The management system shown in Figure 3 will first be referred to, the system having such a functional architecture that the greatest possible benefit will be derived from filtration. A system with this kind of architecture is able to operate e.g. in the system shown in Figure 2. The system comprises a single core part CP and one or more peripheral parts PP1...PPN, which are connected to the core part of the system. In this example, each peripheral part is intended for one user (operator employee) of the system.

The system typically has a dedicated core computer, such as a server SRV (also shown in Figure 2). The core part of the system keeps up an updated model of the state and operation of the telecommunications network TN to be managed, of which a single network element is shown in the figure. As described earlier, the core part uses objects to keep up this model. The core computer SRV is connected to one or several workstations WS1...WSN each including one peripheral part. The functional processes of the system, which run on the core computer and on the workstations, have been marked with ovals. These processes will be described in the following.

An application (indicated with the references

APPLN1...APPLN4 in Figure 3) is a functional system block which allows the user to configure the network and view network events on the screen of the workstation. The application is thus a system part offering the system's services to the user. An individual workstation is therefore also called an application server.

A Session Handler SH is a functional system part that handles a single user's session with the system and forwards messages between the core and the user's applications. The Session Handler can be associated with the initial system log-on window through which a user logs on and establishes a session with the core. The Session Handler starts up various applications according to commands given by the user. When the user logs out, the Session Handler performs the necessary closing-down and deregistering operations to terminate the session (the term registering will be explained later).

A Core Access Process CAP verifies that the Session Handler belongs to a known user on a known host. Having done so, it acts as the gateway to the system core. For each CAP in the core part there is a corresponding Session Handler in the peripheral part.

A Database Process DP controls access to the database DB to store persistent information (master objects of the management system). The main functionality of this process is:

- to convert persistent objects to and from a storable format,

- to store the current state of all persistent objects in the database DB and to keep that database up to date by the use of change and event messages sent to it by a Modelling Process MP of the core part,

- to handle the forwarding of event information

to interested parts of the system.

The Modelling Process MP is responsible for maintaining the currency of the core's model of the network to be managed and dealing with changes to it. The main functions of the Modelling Process are:

- to accept change and event indications from either Interface Process IP or CAP (i.e. either from the network or from the user) and validate them,

- to apply changes to the model if they are valid and to determine the results of these changes,

- to pass the changes to an appropriate Interface Process IP, if the changes require corresponding changes in the network to be managed,

- to generate events based on the changes and events received (e.g. fault events on paths and circuits).

The Interface Process IP is a functional system part that converts data from the external world (data from the network to be managed) into the internal world of the management system. As far as the system user is concerned, he or she just sees a network element NE, like a multiplexer or a cross-connect device, without having to know the manufacturer or the version of the device. Different types of Interface Processes are used for different classes of equipment. The main functionality of each IP is:

- to monitor the network for events and to exchange notifications with the network,

- to translate these events and notifications into equivalent events that are applied to the Modelling Process MP,

- to pass these events and notifications to the Modelling Process,

- to accept changes received from the Modelling Process; translate them into the format of the network

model; and send them to the network and then inform the Modelling Process when all commands relating to each original change have succeeded, failed or timed out.

In fact, the Interface Processes could also be included in the core part, but Figure 3 shows them separate, as they form the interface of the management system with the external world.

A Fault Propagation Agent Process FPAP is a functional system part that works out the actual impact of a received event in the system. In order to be able to do this the FPAP uses information about the object class in question. This information it receives from the Database Process DP.

As the network elements cannot provide all the information needed by the applications, the database DB must store information e.g. about individual network elements and interconnections and interrelations between the network elements, the operations the network elements are capable of performing, and the services provided by the network elements. This information is stored in the form of an object model representing the transmission network TN in terms of objects and their attributes and methods performed on these attributes. Thus, an object is the representation, in the object model, of one of the resources to be managed.

Applications use these objects that form the image of the network to be managed within the system. Therefore, they have to get objects from the core part of the system. An application can also create a new core object to database DB, e.g. as a result of a user command.

Applications also register to receive various events. The registration is passed to the Session

Handler and further to the core part of the system. The registration is stored in the form of a filter F1 in association with the Database Process DP. This filter finds out on the basis of the event type, to which Core Access Processes and to which applications a certain type of event is to be transmitted (i.e. what users are interested in a certain type of event). Events from the core of the system are duplicated in the Session Handler and sent to all applications which have registered to receive them. In this way, all of the user's applications can receive the event at the same time, so that information presented to the user by the different applications is consistent.

In this way the core part of the system knows to which application a certain event has to be transmitted. This also allows unwanted events to be filtered out as soon as possible (in the core) and reduces the bandwidth used between applications and the core part of the system. The core part is thus able to filter events up to the application level. It is also possible to use filters in steps so that the core part knows only up to the level of the Session Handler what event has to be transmitted to the peripheral part. The Session Handler thereafter has a new filter, which indicates in which event a specific application inside the concerned Session Handler is interested. The first alternative, however, is to be preferred in that it does not make the peripheral part (Session Handler) too complicated.

In the management system of Figure 3, part of the objects stored in the core part of the management system are also placed as copies in a cache CH at each Session Handler SH. (Although a cache is a certain kind of memory, in this connection caching means only that a copy of something is kept to get fast access to

it.) Thus, the Session Handler has a copy of any object an application has fetched from the core part of the system. The Session Handler thus also stores object-specific information indicating which one of its applications uses a specific object. This information can be stored e.g. in the form of a table TBL1 shown in Figure 3. The core part correspondingly knows which Session Handler has a copy of a specific master object. This information can be stored in the form of a table TBL2 in association with the Database Process.

At the system start-up, the Session Handler has no copies of objects. When the application needs a copy for the first time, the copy is fetched from the core part. As the transmitted retrieve message includes information specifying the object to be retrieved and the application that needs the copy, the Session Handler is able to keep up a table (TBL1) indicating which application uses a specific object, and the Database Process is able to keep up a table indicating which Session Handler uses a copy of a specific master object.

If an application fetches an object of which the Session Handler already has a copy, the application receives a copy of the object from the Session Handler without having to interact with the core part of the system. When the application ceases using an object, it transmits a message as an indication of this to the core part through the Session Handler, as a result of which the copy is removed from the memory of the Session Handler and the tables are updated accordingly. If the object is needed again, the copy is fetched from the core in the same way as at the system start-up.

When changes are made to an object in an

application, the Session Handler immediately communicates them to all other applications in the session that use said object. (The Session Handler receives information on the change from the application making the change, updates its own copy, and passes the events to all other applications using said object.) These changes are also transmitted through the Session Handler to the core, which reapplies the changes to the master objects to keep the state of the master copies consistent with the state of the copies in the peripheral part.

A system architecture of the type described above is described in the Applicant's parallel patent application GB-A-XXXX, which is referred to for a more detailed description. This application describes e.g. the operation of the different processes more fully.

The present invention relates to the operation of filters (like filter F1 in the above described system) for selecting objects, or for filtering of events, in a management system, which was described briefly above by means of an example. This above-described example of the system is advantageous in the sense that it allows the bandwidth used between the applications and the core part to be reduced efficiently by filtration. Even though selecting objects will be used as an example in the text below, it is to be understood that filtering of objects or events could equally well be used as an example.

According to the invention, selection criteria used to perform filtering are stored assertions about the values of attributes of the objects stored in the system. These assertions are data sets which are here called as Attribute Value Assertions, abbreviated to AVAs. According to the invention, a stored filter is a collection of AVAs linked together by logical

operators (i.e. AND, OR and NOT).

Each AVA is a data set comprising the following three data fields:

- an attribute identifier,
- a value, and
- an operation (e.g. "is equal to", "is greater than", etc.).

Figure 5 shows how filters are used to search for objects from the database. First a filter to be used is created (stage 511) or a previously stored filter is retrieved from the database (stage 512). The first object is then retrieved from the database (stage 513) and filtering is performed (stage 514). Next, it is tested whether the end result is true (stage 515). If so, the object is selected (stage 517), if not, the object is discarded (stage 516). Then it is checked whether there are any objects to be filtered left (stage 518). If so, stage 513 is returned to, if not, the filtering is stopped (stage 519).

Thus, the filter is first fetched to the selection point of the system (if it is not stored there), i.e. to the point where selection is performed. When a filter is applied to an object (stage 514 in Figure 5), the following method steps are performed.

The attribute identifier is first used to get the value of that attribute from the object being filtered. The operation stored in the AVA is then applied to this value and to the value stored in the filter. The operation evaluates to a truth value by performing the operation on the values. This value indicates whether the AVA is true for the value of attribute in the object and the stored value.

If the object does not contain the identified

attribute, then the AVA evaluates to false as the assertion cannot be true for this object.

In the above-described manner a truth value is obtained for each AVA in the filter. Then, the truth values obtained by evaluating each AVA in the filter are combined by the logical operators stored in the filter to yield a truth value for the whole filter. If this final value is true, the object is allowed to pass the filter, i.e. the object is selected. Otherwise it is filtered (not selected).

Thus, within this filtering method the value of the attribute in the object being filtered must be found and must be compared with the value stored in the AVA using the operation stored in the AVA.

In the following the above-described basic principles are explained in a more detailed way by means of an example which is a preferred embodiment where the values of the attributes are stored in a generic form.

In the management system the definition of each type of object describes the attributes within the object. (The attributes of an object are defined in the object model, i.e. they are part of the class that is used to model the object.) The description of each attribute contains the following information:

- an identifier: given an object this identifier can be used to find the value of the attribute within the object.

- a type: this may be a simple type such as String or Integer, or a complex type such as Set of Integers.

- a reference to a ValueMethods object: ValueMethods objects (program blocks) are used to evaluate AVAs. This reference can be implemented in any possible ways, e.g. by means of a pointer word,

i.e. a binary number the value of which corresponds to the object it is pointing to.

In the system, the attributes are stored in the form of the attribute identifiers which contain the type, a unique identifying number forming said identifier and said reference to a ValueMethods object. Values of the attributes are stored elsewhere, as explained later.

To clarify the concept of description of an attribute let us think about a managed network element (like a cross-connection device) as an example. A network element may contain e.g. a Name attribute. This Name attribute would then contain the following information:

- identifier: 1 (if this is the first attribute in the object),
- type: String,
- ValueMethods object: reference (e.g. a pointer) to the String ValueMethods object.

The value of a specific attribute of a specific object can be found by using the object and identifier to find out where in the object the attribute is stored. For example, to find out the name of a network element A:

- the identifier is taken from the attribute (in this example 1),
- the first attribute in A is found in generic form,
- the ValueMethods object is found from the attribute and used to convert the generic form to a String which then gives the name.

As mentioned above, values of attributes are, according to the preferred embodiment of the invention, stored in generic form. This means that all values are stored, regardless of the data type, in

BNSDOCID: <GB 2308777A I >

only one way. In addition to the stored value the system then keeps up information indicating what type of data each memory location contains in order that data could be converted back to specific form.

The storing in generic form can be implemented e.g by using the C++ template mechanism in such a way that there is an abstract base class called e.g. Value and several subclasses, one for each type that is stored in generic form. This is illustrated in figure 4 for integer values, string values and for Boolean values. The following operations may be performed on generic Value objects: (i) copy, (ii) store to a file, (iii) retrieve from a file, (iv) apply a test (where the test applied is one of the operations stored in AVAs). For performing the test, a subroutine called an ApplyTest Method is associated to each ValueMethods object. The ApplyTest Method performs the comparison between the generic values. Each ValueMethods object knows the type of the value of the attribute, i.e the type of the associated ApplyTest Method corresponds to that type.

In order to find the value of an attribute by means of the object and the attribute identifier, the following two steps are undertaken. First, a checking is performed to see if the object has the attribute we are interested in. If the object does not have the attribute, an error is reported at this stage. (The attribute may be defined either in the object or in one of its ancestors.) Secondly, a member pointer is combined, in a manner known as such, with the pointer to the object to find the value of the attribute. (This known member pointer, which is an offset into an object, is contained in the attribute identifier. When combined with a pointer to an object the member pointer points to an individual data member or method

of an object.)

ValueMethods objects perform a number of operations for the attribute type. There is one ValueMethods object in the system for every supported type. For example there may be three String attributes in the system for a network element: name, manufacturer and physical location. These three attributes then all share a single String ValueMethods object.

ValueMethods objects perform the following services in the system:

- a range of supported operations: the operations supported will depend on the attribute type that this ValueMethods object is managing. A String ValueMethods object would support isEqual, isNotEqual, startsWith, endsWith and contains; whereas an Integer ValueMethods object would support isEqual, isNotEqual, isGreaterThan, isLessThan, etc.

- conversion of a value from a generic form to a specific data type.

As mentioned above, all values in the management system are preferably held in a generic form. This generic form allows the system to store values without knowing their type. The generic form also provides an efficient way of implementing the test in which the logical operators stored in the filter are used. The ValueMethods object is able to convert from this generic type back to a specific type, for example from a generic representation of a String to a Specific String.

To give an example, the name attribute introduced above contains a reference to a String ValueMethods object. The String ValueMethods object will support the following operations:

- range of operations: isEqual, isNotequal,

startWith, endsWith, contains.

- conversion to String: a generic value can be converted to a string.

As already mentioned, each AVA comprises an attribute, an operation and a value. The operation should be one of the operations supported by the ValueMethods object referenced by the attribute. If an illegal operation is used, e.g. a startWith operation on an Integer, the AVA will evaluate to False. The value stored in the AVA is held in generic format as mentioned above. This value will then be compared to the value of the specified attribute using the specified operator. (The comparison is made using said generic values.) To give an example, let us imagine that there is a need to find all network elements whose name begins with "SXC". The AVA for this purpose would be created with the following fields:

- attribute: the Name attribute (as described in the above example),

- operation: startsWith, and

- value: the generic representation of the string "SXC".

Figure 6 illustrates evaluating a single AVA. First the attribute is fetched from the AVA (stage 611). Then, it is tested if the object has this attribute (612). If no, the result of the AVA is false (stage 619). If yes, stage 613 is entered, where the ValueMethods object for the attribute in the object is searched for. After the ValueMethods object has been found, the ApplyTest Method of the ValueMethods object is called. Then it is tested (stage 615) if an ApplyTest Method of the correct type can be found. If the types of the ApplyTest Method found and the values to be compared are different, the AVA is false (stage 619). If an ApplyTest Method of the correct type was

NSDOCID: <GB___2308777A__I_>

found, it is tested at the next stage whether the test succeeds. If yes, the AVA is true, if no, the AVA is false. The ApplyTest Method is a generic operation (comparing generic values) but at the same time a specific operation for the particular types of values that are compared with each other.

In the following the method according to the present invention is illustrated by referring to the example of figures 7 and 8. In this example a filter is applied to an object which it matches (object 1, figure 7) and to one which it does not (object 2, figure 8). Further, object 2 does not contain one of the attributes which the filter attempts to filter on.

The filter stored at the selection point of the network is defined functionally as follows: (ExternalName, isEqual, "name") AND (FaultState, isGreaterThan, 1). This means that the filter asserts that the object has an attribute ExternalName which has a value of "name" and that the object also has an attribute FaultState which has a value greater than one.

Object 1 is filtered as follows (figure 7):

1. Node A is evaluated which recursively evaluates its two child nodes.

2. Node B is evaluated.

2a. The value of the ExternalName attribute is extracted from object 1.

2b. The operation isEqual is applied to the value "name" and the value obtained at stage 2a from object 1. This yields "True" as the two values are equal.

3. Node C is evaluated.

3a. The value of the FaultState attribute is extracted from object 1.

3b. The operation isGreaterThan is applied to

the value 1 and the value obtained at stage 3a from object 1. This yields True as 3 is greater than 1.

4. The whole filter evaluates to (True AND True) which is True and the object is passed by the filter.

Object 2 is filtered as follows (figure 8):

1. Node A is evaluated which recursively evaluates its two child nodes.

2. Node B is evaluated.

2a. The value of the ExternalName attribute is extracted from object 2. There is no ExternalName attribute so False is returned immediately.

3. Node C is evaluated as (False AND anything) which is False, so the whole filter evaluates to False. Thus, the object is not passed by the filter.

Though the invention has been described above referring to the example of the attached drawing, it is obvious that the invention is not restricted to that, but it can be modified in many ways within the scope of the inventive idea presented above and in the attached claims. As mentioned above, the management system includes a variety of objects, representing both logical and physical entities. In the enclosed claims all these objects are are set forth as "the objects that relate to the network elements to be managed", i.e. objects that represent the managed environment.

Claims:

1. A method for managing a telecommunications network, said telecommunications network comprising several network elements to be managed by the system,

said system comprising a management centre having at least one workstation accomplishing a man-machine interface and allowing a manager to control the system, and the system being capable of providing the manager with information on the network, said at least one workstation being connected to a database including information about the managed network, said information being in the form of objects that relate to the network elements to be managed and in the form of references between the different objects, said references indicating the dependencies between the objects, whereby said management centre is connected to said network elements by data communication links such that the manager can initiate an operation on and receive information from a managed item of the network, in which method objects are selected at separate selection points of the system, **characterized in that** separate sets of data with logical operators linking the sets together are sent to a selection point of the system to be stored there, each set of data comprising an attribute identifier, a value and an operation, and the selection is performed at the selection point so that an intermediate result is first derived for each set at the selection point by

- using said attribute identifier to get the value of the same attribute from the object being selected, and

- applying said operation to this value and the value stored at the selection point to obtain said intermediate result, and

the intermediate results obtained are combined by the stored logical operators to get a final result, and

the final selection of the object is made on the basis of the final result.

2. A method according to claim 1, **characterized in that** the values of the attributes are stored in generic form which is independent of the data type in question.

3. A method according to claim 2, **characterized in that** said separate sets of data include a reference to an object which is able to (a) perform operations on the values of attributes generically, (b) convert said generic value back to a specific data type.

4. A method for managing a telecommunications network, said telecommunications network comprising several network elements to be managed by the system,

said system comprising a management centre having at least one workstation accomplishing a man-machine interface and allowing a manager to control the system, and the system being capable of providing the manager with information on the network, said at least one workstation being connected to a database including information about the managed network, said information being in the form of objects that relate to the network elements to be managed and in the form of references between the different objects, said references indicating the dependencies between the objects, whereby said management centre is connected to said network elements by data communication links such that the manager can initiate an operation on and receive information from a managed item of the network, in which method events are filtered at separate filtering points of the system, **characterized in that** separate sets of data with logical operators

linking the sets together are sent to a filtering point of the system to be stored there, each set of data comprising an attribute identifier, a value and an operation, and the filtering is performed at the filtering point so that an intermediate result is first derived for each set at the filtering point by

- using said attribute identifier to get the value of the same attribute from the event being filtered, and

- applying said operation to this value and the value stored at the filtering point to obtain said intermediate result, and

the intermediate results obtained are combined by the stored logical operators to get a final result, and

the filtering of the event is made on the basis of the final result.

5. A method according to claim 4, **characterized in that** the values of the attributes are stored in generic form which is independent of the data type in question, whereby said operation to said value and the value stored at the selection point is performed using said generic values.

6. A method according to claim 4, **characterized in that** said separate sets of data include a reference to an object which is able to convert said generic value back to a specific data type.

7. A method for managing a telecommunications network, substantially as hereinbefore described with reference to Figs 3 to 8 of the accompanying drawings.

The Patent Office

28

Application No: GB 9526600.3
Claims searched: 1 to 7

Examiner: M J Billing
Date of search: 26 April 1996

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): H4K KF42.

Int Cl (Ed.6): H04Q 3/00.

Other: ONLINE : WPI.

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	WO94/06232A1 (ERICSSON) - Abstract	1,4

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

An Executive Agency of the Department of Trade and Industry